**信息隐藏实验报告**

**实验三**

**扩展技术**

****

**学 号 1750844**

**姓 名 周展田**

**专 业 计算机科学与技术**

**同组成员 1754259袁瑞毅**

**授课老师 钟计东**

**一、实验内容**

在Lena图中嵌入一个64×64(共4096位) 的Logo，需要使用STDM方法嵌入在DCT系数中。

性能比较

1.（不同噪声强度）：在相同的量化步 长下，不同强度的噪声攻击下(可以用图像处 理软件加入噪声)系统性能的变化的曲线

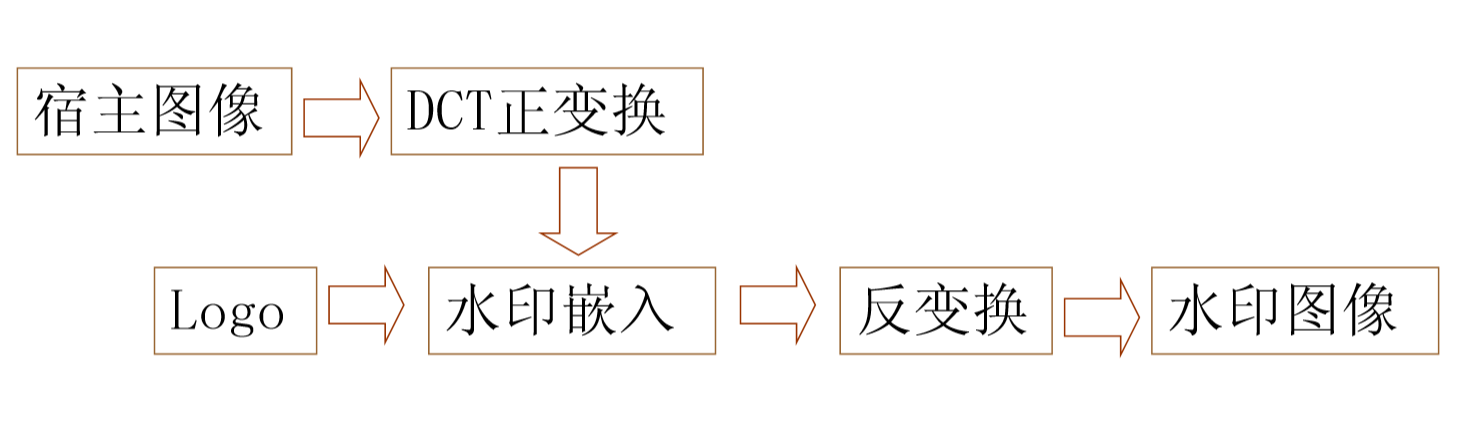
2.在相同的量化步长下，不同强度的 JPEG压缩(可以用图像处理软件进行压缩)下系 统性能的变化的曲线

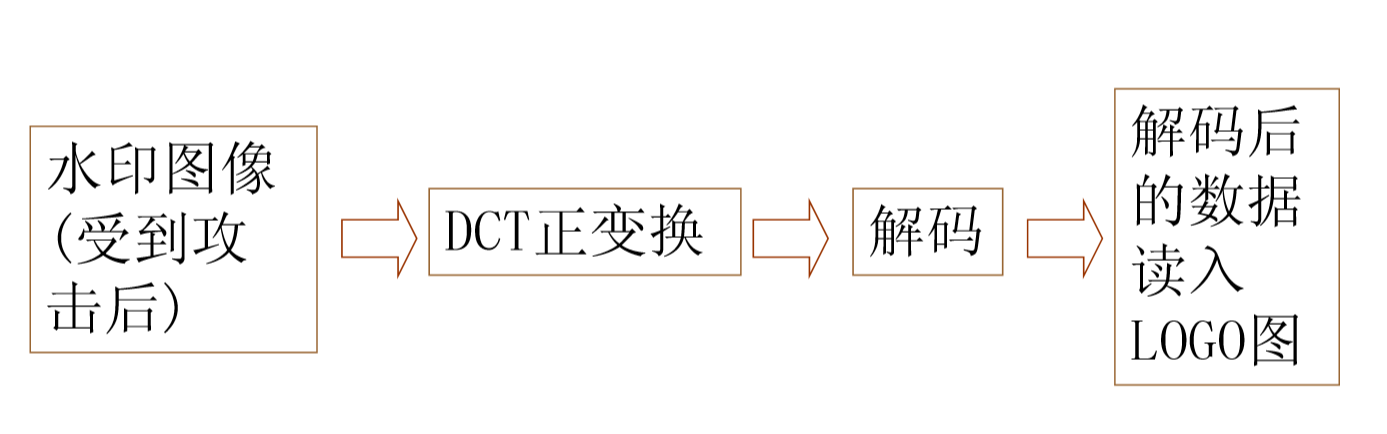
3.不同量化步长下，相同噪声攻击强 度的性能变化曲线

**二、实验原理**

1.STDM

(1)水印的嵌入步骤和解码步骤：



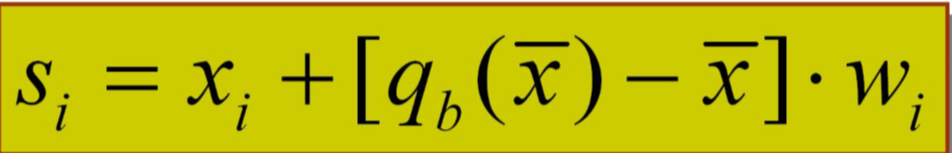


(2)读取图片：嵌入信息为一二进制 Logo图，黑色用0表示，白色用1表示。为了方便嵌入，将b = 0用b = −1表示，此时 b就代表了扩频算法中的b。在读取二进制图片时要按byte读取，对应于 C语言中的 unsigned char，此时一个 byte表示了8个像素，因此读取后还要提取其中的每一个 bit。

(3)DCT变换：DCT是离散余弦变换(Discrete Cosine Transform)，如果原始数据用矩阵X表示，变换后的数据用Y表示，那么DCT变换就是这样的一种变换： Y ＝ TXT’，T 是一个正交矩阵满足 TT’ ＝ E，其中 T’是 T 的转置矩阵。反变换是 X = T’YT 。因此实验中首先对图像(像素)进行变换，求得变换后的 DCT系数，然后用ASS 在中频系数嵌入信息，嵌入后还需要反DCT变换。

(4)嵌入原理：

代码中设置d[0]=-Δ/4，d[1]=Δ/4

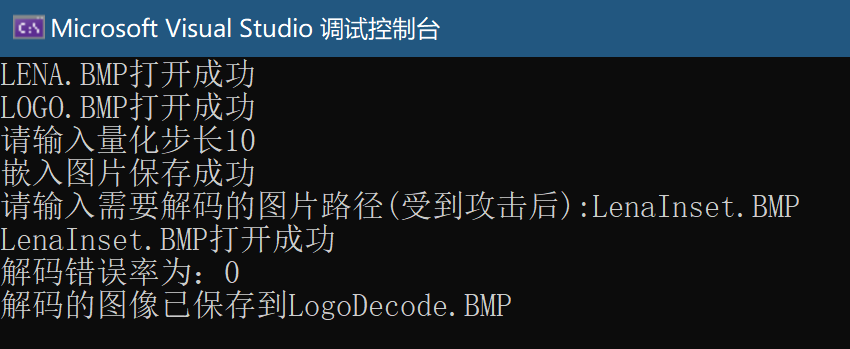


(5)解码原理：



**三、实验结果**

1.未受到攻击时，输入量化步长为10，解码得到的错误率为0



可以看到解码得到的logo图片于原来的logo一致

2.性能比较：

关于噪声的添加，我们选择了高斯白噪声作为我们的噪声种类。通过如下的matlab代码向BMP图片中加入噪声。



①不同噪声强度下的性能比较：首先我对量化步长为10的BMP图片进行高斯白噪声的添加：



每次强度递增10，得到六张图片：













可以看到，加入噪声后对图片清晰度的影响是十分明显的。

解码得到的LOGO也有很明显的变化：





（后续LOGO由于噪声强度较大，已经不能辨认不再贴出。）

然后再比较相同量化步长不同噪声强度下的错误率：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 噪声强度非百分比 | 0 | 10 | 20 | 30 | 40 | 50 |
| 错误率 | 0 | 0.148 | 0.434 | 0.513 | 0.495 | 0.459 |

根据以上数据，以噪声强度变化为横坐标，以错误率为纵坐标画出折线图如下：

可以看出随着噪声强度的不断增大，大约在30左右，错误率达到了峰值，大约0.5左右。之后随着噪声强度的继续增大，错误率开始呈现下降趋势。

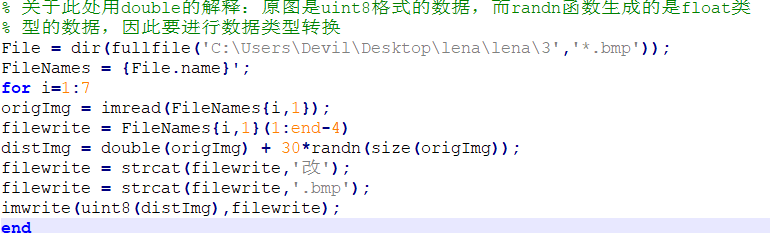
② 在相同的量化步长下，不同强度压缩下系统性能的变化：

这部分的实验不是很成功，在下面遇到的问题中做详细描述。

③ 不同量化步长下，相同噪声攻击强度的性能变化曲线：

为了比较不同的量化步长，对最终系统性能的影响，我分别选用了10、15、30、50、100、200、1000.99共7个不同的量化步长，添加相同强度的高斯白噪声，比较最后的错误率。

添加高斯白噪声的MATLAB代码为：



高斯白噪声对图片的影响在①中已经展示过，这里不再重复展示。

最终得到的不同量化步长，加以相同强度噪声的错误率如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 量化步长 | 10 | 15 | 30 | 50 | 100 | 200 | 1000.99 |
| 错误率 | 0.480 | 0.423 | 0.151 | 0.013 | 0 | 0 | 0 |

根据上表的数据，画出对应的折线图为：

通过上图能够看出，在相同的噪声强度下，先随着量化步长的逐渐增大而变小。但是当量化步长达到一定程度时（约为100左右），噪声强度对解码的错误率不发生影响，错误率始终为0。

**四、实验中遇到的问题**

1.在数据嵌入时没有遇到太大的问题，只需要将嵌入与解码的两个函数进行一定的修改即可，同时注意在解码时也需要用到量化步长。

2.性能分析

性能分析中遇到的问题主要是在BMP格式的图片压缩上。首先我尝试用图像处理软件来进行BMP图片的压缩。在百度上找到了找到了“迅捷压缩”的工具，他虽然支持BMP格式的压缩，但是不能控制压缩的强度，而且下载压缩后的图片还要付费。然后我尝试用PS对BMP图片进行压缩，想通过BMP图片导出为JPG格式时降低其品质，再将该JPG文件保存成BMP格式的方式进行BMP文件的压缩，但是发现这种方法无法保持BMP 文件深度的一致。使用老师给出的深度为8位的BMP文件导出为JPG文件之后，深度会自动调整为24位，再次对JPG进行压缩的时候只能选择生成BMP的深度为16或24位，并没有8位的选项。所以无法实现压缩的目的。最后一种方式我尝试用GIMP软件进行压缩，是由梁育豪同学推荐的一款软件，但是经下载尝试之后发现无法启动就会闪退。使用JPEG算法压缩BMP的方法也没有找到。所以第二个小实验没有继续进行下去。

**五、分工说明**

周展田： 编写程序，实现嵌入和解码

袁瑞毅： 进行性能分析

**六、参考资料**

BMP图像的读取代码参考于：

<https://blog.csdn.net/lyy289065406/article/details/6717679>

DCT变换的代码参考于：

<https://blog.csdn.net/lwfcgz/article/details/8040550>